Manuscritos digitales en Física

Alberto Corbi

Los sistemas de composición de documentos para la enseñanza de la Física van desde TeX/LaTeX hasta el moderno Typst, pasando por lenguajes de marcado ligero, como Markdown. El denominador común de todos ellos es el texto plano y la separación entre estilo y contenido. Por otro lado, la programación letrada y la nube facilitan la confección de materiales educativos más prácticos y su distribución a estudiantes, colegas, etc.

A pesar de que la enseñanza y divulgación de la ciencia está cada vez más influenciada por animaciones interactivas, dispositivas, vídeos explicativos, simulaciones, *apps*, *playgrounds*, e incluso por películas y series de televisión del género de la ciencia ficción *hard*, qué duda cabe que, en el ámbito de aprendizaje de la Física, el medio de transmisión de conocimiento por excelencia es el manuscrito. Ciertamente, el texto científico posee sus idiosincrasias y características propias, tales como la aparición de expresiones matemáticas, cuadros y gráficos sinópticos, datos tabulados, código informático, citas bibliográficas, autorreferencias (a figuras, tablas y secciones), etc. La Física, si cabe, redobla aún más estas exigencias con el uso de sus propios elementos (diagramas de Feynman, croquis de montajes experimentales, esquemáticos de electrónica, símbolos matemáticos específicos, etc.).

La llegada de TeX/LaTeX

Hasta hace unas pocas décadas, las particularidades que acabamos de mencionar solo podían ser satisfechas por entornos editoriales profesionales. Sin embargo, todo cambió con la llegada de TeX a finales de los setenta: un revolucionario sistema de edición creado por Donald Knuth [1]. TeX consistía en un lenguaje de *macros* (rutinas de código que se *inflan* cuando son analizadas por un programa llamado *procesador*) que era capaz de producir letras correctamente dibujadas mediante innovadoras curvas de Bézier, perfiladas con hipnótica belleza y apiladas una tras otra con precisión quirúrgica. Además, por primera vez, era posible codificar expresiones matemáticas mediante un lenguaje de programación *ad hoc* cuya salida era perfección visual en estado puro.

Al estar basado en macros, TeX podía ser fácilmente expandible para aumentar sus funcionalidades y/o alterar los resultados producidos, y muchos de sus usuarios se lanzaron a crear sus propias colecciones de macros. De todas ellas, la que tuvo un éxito arrollador fue la del matemático Leslie Lamport. Su ampliación de TeX, la archiconocida LaTeX (1984) es, con pequeños cambios y mejoras atómicas, la que se sigue utilizando 40 años después. TeX y LaTeX son la definición perfecta de *software* bien hecho. Su código conjunto ha sobrevivido intacto a golpes de estado, 7 administraciones federales americanas, 5 papas y 12 misiones exitosas (no tripuladas) a Marte.

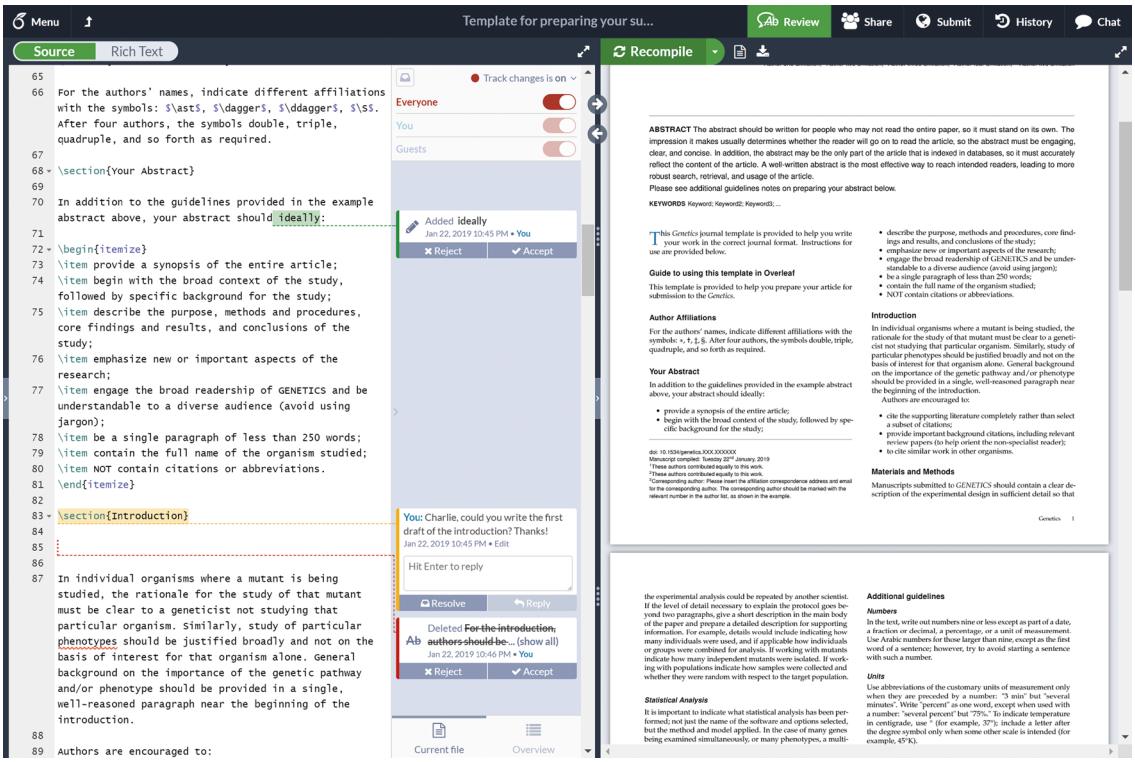
LaTeX fue además el primer entorno de composición de documentos que incidió con firmeza en el hecho de separar *estilo* de *contenido*. Esta distinción es muy importante en la comunicación en Física, ya que lo primordial es la idea/descubrimiento a transmitir (contenido + estructura), y en segundo lugar está siempre el formato (estilo). LaTeX consigue esto mediante los así llamados *ficheros de estilo* (.sty) y de *clase* (.cls), aunque en realidad, estos no son más que colecciones de macros en lenguaje TeX.

La autoedición

También a mediados/finales de los años ochenta, pero con menor relevancia en ciencia, surgió la efervescente industria de la autoedición (*desktop publishing*), principalmente de la mano de Apple, el Macintosh y la impresora LaserWriter. Por primera vez en la historia de la informática (y de la humanidad),



Fig. 1. El Mac y la impresora LaserWriter (fuente: Apple).



se había democratizado la producción de documentos impresos con calidad fotográfica (fig. 1).

En años siguientes, esta difusión de la autoedición siguió su curso gracias a la implantación generalizada de los ordenadores personales y la aparición de una miríada de programas llamados procesadores de texto (desde Microsoft Word 1.0, Word Perfect y MacWrite hasta las versiones actuales de Office 365, LibreOffice y Apple Pages). Sin embargo, la producción de textos científicos de calidad siguió delegando en LaTeX la mayor parte de su cuota. Esto era debido principalmente a la necesidad de redacción de expresiones matemáticas de calidad, tarea en la cual, LaTeX no tenía rival. Estimaciones conservadoras afirman que un 30 % de la producción de textos científicos aún se realiza en este entorno. LaTeX también ha sido portado a la nube por empresas como Overleaf (fig. 2).

Ciertamente, los entornos ofimáticos antes mencionados han mejorado mucho en los últimos años y ya han cerrado en buena medida el *gap* con LaTeX. Sin embargo, siguen adoleciendo de un defecto importante para la confección de manuscritos científico-técnicos: no separan contenido de estilo de la manera tan tajante a como lo hace LaTeX (u otros enfoques que luego veremos).

Es interesante destacar en este punto el proyecto LyX [2], lanzado por primera vez en 1995. LyX es un procesador de documentos que combina la potencia de LaTeX con una interfaz de usuario intuitiva y orientada al WYSIWYM (*What You See Is What You Mean*). Este entorno ha ido ganando popularidad entre académicos y profesionales que

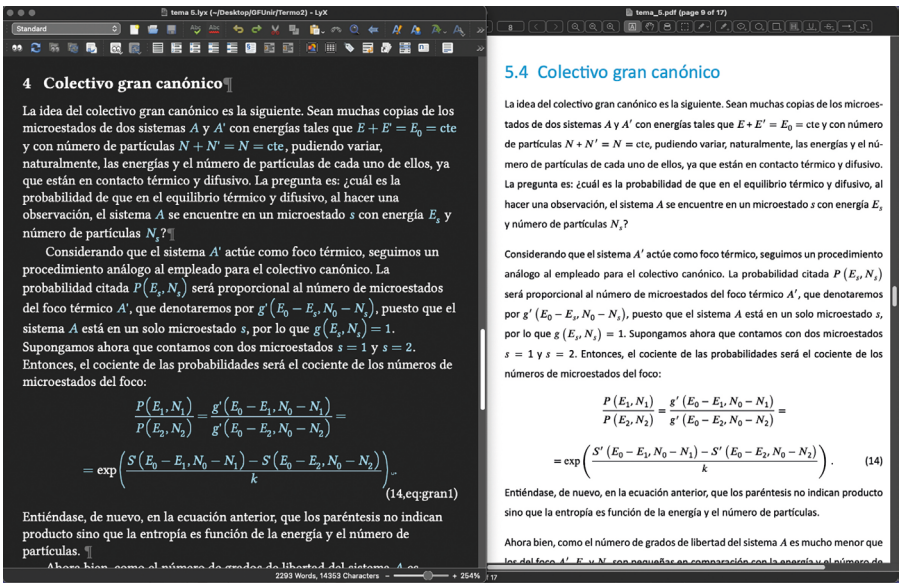


Fig. 2. Un documento LaTeX en la nube con Overleaf.

Fig. 3. Procesador de textos LyX junto a su salida en PDF (el documento se compila con XeLaTeX, una variante de TeX que admite más estándares de tipos de letra).

buscan una forma eficiente, a la par que amigable, de trabajar con LaTeX (fig. 3). LyX nos presenta un documento editable y con un estilo aproximado a cómo quedará finalmente cuando sea *compilado* por un motor LaTeX (y obtengamos un PDF). Por esta razón, LyX no es un entorno WYSIWYG (*What You See Is What You Get*) propiamente dicho, pero sí un compromiso entre un entorno TeX tradicional y un *software* clásico de ofimática.

La Web y el MathML

Las tornas comenzaron a girar en 1998 con la llegada de la web, su lenguaje de descripción de páginas (HTML) en texto plano, su organismo de *vigilancia* (W3C) y la estandarización de la escritura matemática gracias a una sintaxis de marcado basado en XML. MathML (*Mathematical Markup Language*) permite la presentación y comunicación de contenido matemático de una manera comprensible tanto para humanos como para máquinas [3]. Se divide en dos partes principales: *presentation*, que se centra en la presentación visual de las expresiones matemáticas, y *content*, que se enfoca en la representación semántica de las estructuras matemáticas, con el objetivo, entro otros, de facilitar la accesibilidad [4] para lectores con dificultades de visión.

A pesar de las buenas intenciones de MathML, el tándem TeX + LaTeX lo había hecho tan bien desde sus orígenes, que muy pocos autores realizaron la transición a los estándares web para la producción de *papers* científicos. La culpa de esta escasa implantación también la ha tenido el propio marcado XML, el cual resulta farragoso (fig. 4).

Pero qué duda cabía que MathML y la web habían venido para quedarse. Por esta razón, algunos ingenieros comenzaron a preguntarse: ¿por qué no fusionamos lo mejor de ambos mundos? De esta manera nacieron proyectos como MathJax [5] y KaTeX. Ambos hacen una cosa muy sencilla: convierten expresiones matemáticas redactadas con sintaxis LaTeX (más conocida) a MathML. A pesar de ello, el resto de los componentes de un artículo o texto científico (citas, seccionado, referencias cruzadas, etc.) quedaban por implantar en los estándares web (algo en lo que LaTeX se había vuelto muy eficiente).

Lenguajes de marcado ligero

Nuevamente, las cosas volvieron a tambalearse con la aparición de *subsets* de HTML, es decir: lenguajes de descripción de páginas web pero que se centraban más en el contenido, deshaciéndose de la *morralla sintáctica*. Y no un contenido cualquiera: escritos de índole técnico-científico donde las expresiones matemáticas se podían seguir redactando en el archiconocido LaTeX. El mayor exponente de estos nuevos estándares industriales es Markdown, descrito originalmente en 2004 por el periodista John Gruber y el tristemente fallecido activista Aaron Swartz. Con Markdown es

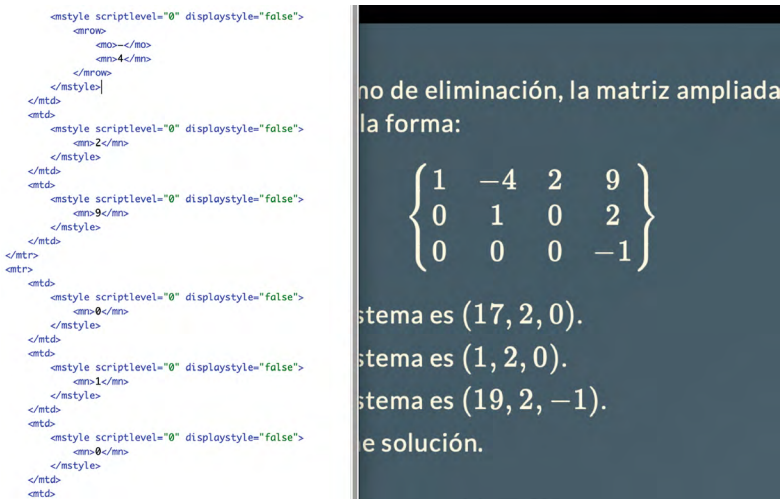


Fig. 4. MathML, el lenguaje matemático propuesto por el W3C. En este caso se muestra la codificación de una matriz (izquierda) y su representación en una página web (derecha).

posible redactar documentos bien seccionados, legibles, de manera veloz y, nuevamente, en texto plano. Además, mediante su emparejamiento con herramientas de procesado, es posible incluir citas, autoreferencias, etc., es decir, todo lo que necesita un texto científico prototípico. La salida por excelencia de un documento redactado en Markdown es una página web (HTML), pero gracias a procesadores como Pandoc [6], es posible generar su contrapartida LaTeX, DOCX, LibreOffice, PDF y otros muchos formatos. En resumen: LaTeX tiene un serio rival con una sintaxis renovada, simple y un ecosistema de herramientas joven y multiplataforma: escritorio, móvil o en la nube (Dillinger). Además de Markdown (fig. 5), existen otros así llamados *lenguajes de marcado ligero* (AsciiDoc, reStructuredText, Org Mode [7], etc.). Su denominador común es una sintaxis sencilla para separar la estructura y estilo en documentos expresados en mero texto plano.

La programación literaria

La idea original de Donal Knuth con TeX no era simplemente un sistema de composición de docu-

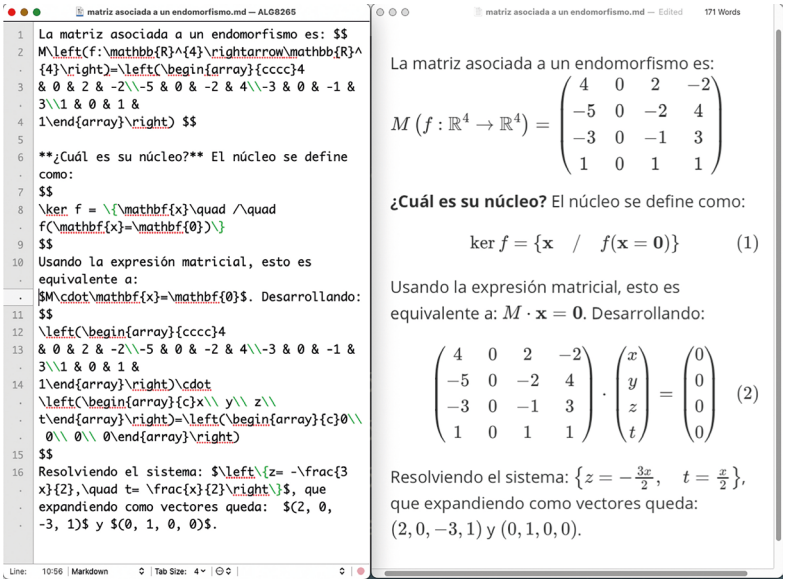
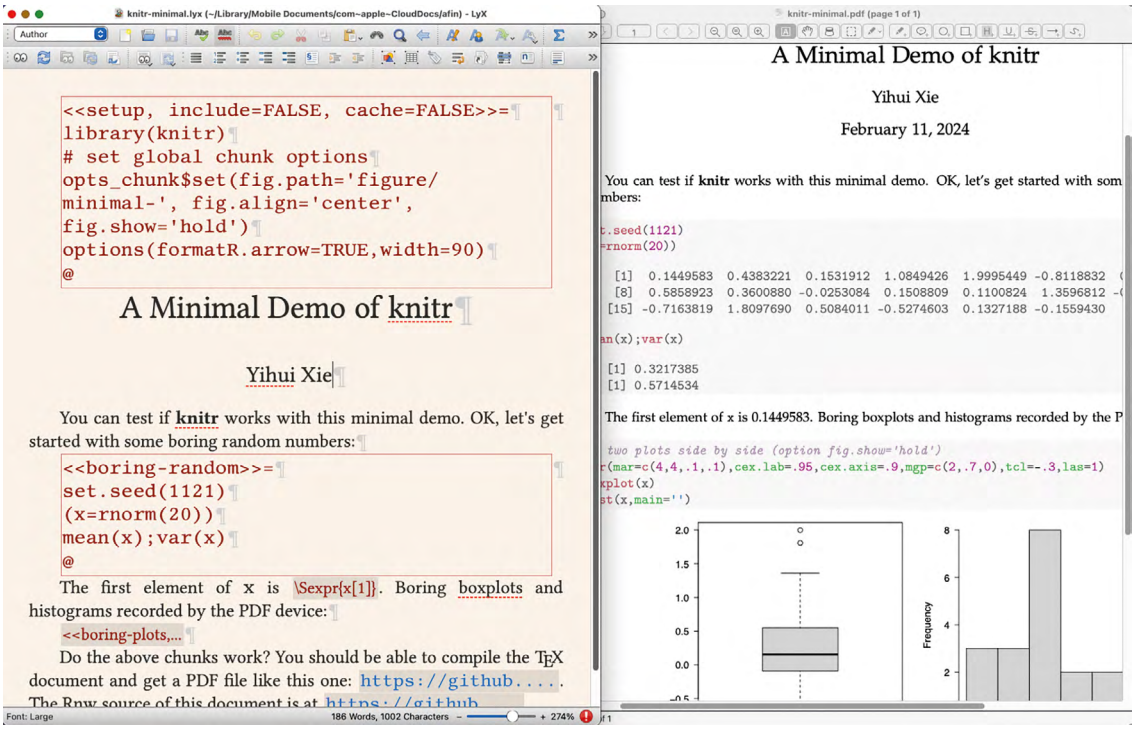


Fig. 5. Documento Markdown y su traducción a PDF mediante el procesador Pandoc.



mentos, sino una propuesta para una armoniosa interacción entre segmentos de código *vivo* (ejecutados a voluntad del usuario), texto explicativo convenientemente estilado y resultados [8]. Para Knuth, un texto científico debía ser una suerte de *diálogo computacional* con el lector y a este enfoque lo llamó *programación letrada* o *literaria*. Dicho de otra manera, un programa debía ser más bien una narración, un *relato*.

Desafortunadamente, esta loable idea cayó en el olvido y solo tuvo contadas expresiones exitosas. Quizás la más representativa durante la década de los noventa y la primera década del siglo XXI, ha sido el formato *notebook* de Mathematica (Wolfram). Un *notebook* de este sistema contiene bloques (o celdas) con texto formateado, código en el lenguaje Wolfram [9] y resultados generados cuando las celdas de código son evaluadas por el lector. Los bloques se pueden mover y anidar libremente, dotando además de estructura al documento final, el cual se parece a una suerte de *cuaderno digital* o incluso al embrión de un artículo de investigación. De hecho, hay investigadores que abogan por la extinción del *paper científico* en favor de los *notebooks* [10].

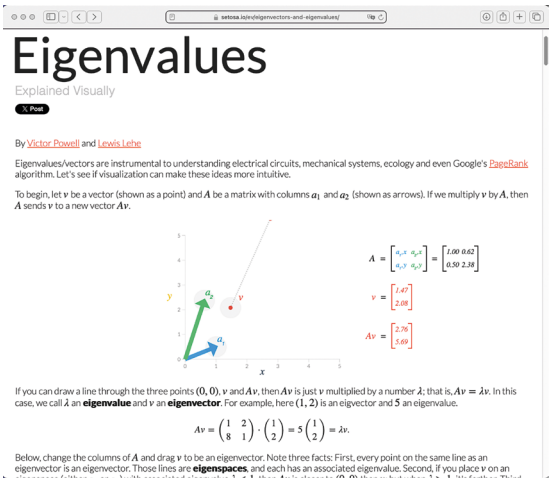


Fig. 7. Explicación explorable relacionada con autovectores y autovalores.

De manera inesperada, la programación letrada ha vuelto a la palestra en los últimos años gracias a proyectos como Jupyter, los contenedores virtuales (Docker [11]), la nube y las propias tecnologías web. Hoy en día, el uso y despliegue de estos documentos es masivo, generalizado y fácil para su uso en educación [12]. La reciente pandemia de COVID-19 ha sido un claro escenario donde se ha podido comprobar la utilidad y salud de estos, ya que gran parte de los materiales educativos en Física se *subieron* a sitios como MyBinder [13], que permite lanzar *notebooks* de tipo Jupyter con un simple enlace web. Además de Jupyter existen otras arquitecturas de *notebooks* (fig. 6) como Quarto [14] o Knitr [15]. Si bien estas tecnologías de *notebooks* aparecen típicamente asociadas a un lenguaje de programación concreto (Jupyter + Python, Knitr + R, etc.), hoy en día, todas ellas son bastante agnósticas en cuanto a *núcleo de computación* [16].

El enfoque literario no solo otorga capacidad narrativa al boletín de ejercicios o al guion de laboratorio (o cualquier otro material didáctico), sino cohesión entre experimento (aunque este sea una simple secuencia de cálculos) y la descripción asociada. En un documento letrado, el escrito final es actualizado y *calculado* de manera dinámica en cada compilación (en contraste con el enfoque tradicional *estático*).

Reactividad y *tareas explorables*

En torno a 2010, el científico de la computación Bret Victor comenzó a abogar por enfoques más intuitivos y visuales para docencia de la ciencia.

Es el padre de los conceptos de *medio dinámico* y de *programación reactiva*. Este último es un paradigma de programación que se centra en la propagación automática de los cambios de estado en cualquier parte de un programa (o texto redactado) sin necesidad de código imperativo.

La reactividad está íntimamente conectada con los *documentos explorables* [17]. En lugar de presentar información de manera estática, estos entornos permiten a los estudiantes y profesores interactuar directamente con el contenido (presentado en forma de página convencional), manipular variables y observar resultados en tiempo real. Victor aboga por herramientas que brinden retroalimentación inmediata, facilitando un aprendizaje más activo. Estos materiales docentes *orgánicos* pueden incorporar también simulaciones interactivas y gráficos dinámicos que permiten al estudiante experimentar directamente con los principios enseñados en clase (fig. 7).

Cuando un *notebook* combina narración y *feedback inmediato* se consigue una experiencia docente mucho más compacta, eficiente y significativa. Actualmente existen varias opciones que un estudiante de Física puede usar. Quizás, la más exitosa es Observable [18], con el que podemos crear notebooks inmersivos de alta calidad (fig. 8).

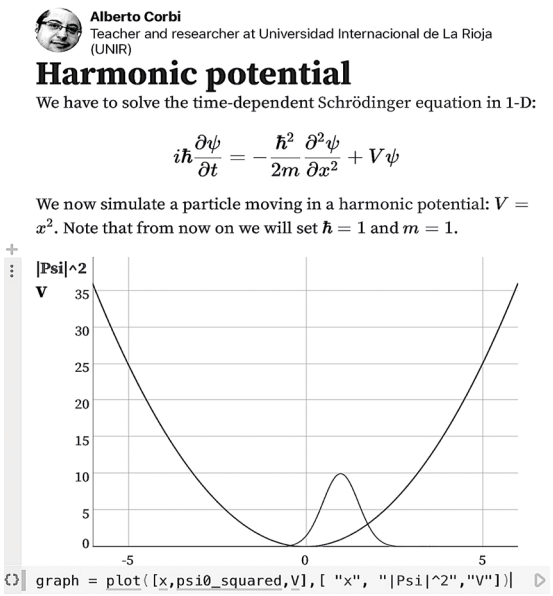


Fig. 8. Ejemplo de notebook basado en el enfoque reactivo de Observable.

Documentación y tareas en la docencia de la Física

Una vez realizado este recorrido histórico por los sistemas de documentación digital en ciencia, veamos cuáles son los principales retos y recomendaciones para su uso en la enseñanza de la disciplina que nos atañe.

Para la docencia en Física y la elaboración de tareas, tanto en entornos presenciales como a distancia, deberían de fomentarse los estándares informáticos y académicos resumidos anteriormente. En cierto sentido se puede constatar que así es y, muchas universidades ya enfatizan, por ejemplo, el uso de LaTeX entre el alumnado de primeros cursos. Sin embargo, la experiencia común en docencia es que este sistema es algo complejo para pequeñas tareas como la elaboración de boletines de problemas o la entrega de simples guiones de laboratorio. LaTeX tampoco es interactivo, lo cual le resta atractivo en un escenario académico cada vez más multimedia.

Un Lamborghini por un camino rural

Cuando un estudiante de Física se sumerge en LaTeX por primera vez y comienza a disfrutar de los primeros resultados impresos, tiende a pensar que *ya está todo hecho* solo por la mera razón de hacer uso del sistema edición más empleado por su querido gremio. Se tiene la ingenua sensación de estar a bordo del *Lamborghini de la autoedición de textos científicos*. Ciertamente, la gran mayoría de entornos LaTeX vienen preparados para producir artículos haciendo uso de las tipografías y estilos por defecto. Es decir: se pasa por alto por completo, entre otras cosas, la imagen institucional (tipo de letra, márgenes, interlineado, sangría, etc.). Además, si bien LaTeX viene de serie con cualidades para producir bellos documentos desde la primera compilación, muchas opciones dependen de la formación del estudiante en expresión escrita y matemática. LaTeX tampoco es un *paradigma mágico* para la generación de gráficos atractivos y vectoriales, ni para la visualización de datos y resultados. Tampoco permite la comunicación con entornos computacionales de terceros. El resultado de un documento TeX elaborado por un estudiante de primeros cursos suele ser un PDF *inerte*, incorrectamente formateado y con errores de estilo importantes. Es decir, es como si hubiéramos conducido un deportivo de lujo (un Lamborghini, por ejemplo) por un camino rural.

Por esta razón, es necesario incentivar, ya desde educación secundaria, el uso de marcados más simples, modernos, abiertos y legibles. Markdown es sin duda el rey en este escenario. Además, al estar basado en estándares web, es relativamente sencillo adecuar un documento para respetar las reglas de estilo de la institución o editorial. Asimismo, puede ser copiado a servidores y distribuido cómodamente como un enlace. Pero la ventaja más interesante de todas es que un escrito científico compuesto mediante cualquier lenguaje de marcado ligero, al estar basado en texto plano, ocupa muy poco en disco y requiere un ancho de banda ínfimo para su transmisión a través de la red.

URLs como material docente y entregables

La utilización de la nube para la elaboración de contenidos didácticos o la remisión de tareas en línea ofrece una serie de ventajas significativas en términos de flexibilidad, accesibilidad y colaboración. En primer lugar, la nube permite a los creadores de contenido acceder a sus archivos desde cualquier lugar y en cualquier momento, eliminando las restricciones

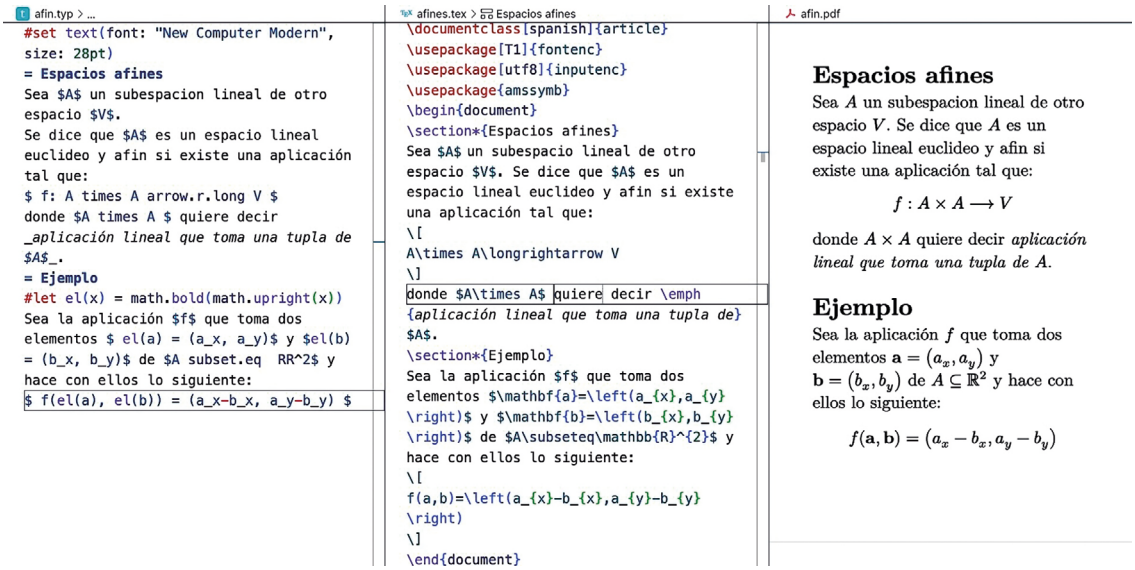


Fig. 9. Typst, la auténtica alternativa a LaTeX. En el panel de la izquierda tenemos el código Typst, en el del centro tenemos el código LaTeX y en el de la derecha el resultado (muy parecido para ambos) en PDF.

geográficas y fomentando la productividad en entornos remotos. Por otro lado, varios estudiantes pueden editar y revisar documentos simultáneamente.

Pero lo mejor de todo es que siempre tenemos a nuestra disposición un enlace o URL, que los alumnos pueden mandar al profesor con la tarea correspondiente. Estos recursos pueden ser perfectamente notebooks, documentos TeX o Markdown, archivos ofimáticos, código informático, etc.

El documento escrito como *proyecto*

Otra gran utilidad de trabajar en el marco de ficheros de texto plano, marcado sencillo y con separación de estilo y contenido, es que uno puede empezar a *colaborar consigo mismo*. Esto puede parecer una tontería, pero no lo es. Cuando un académico o estudiante adopta este enfoque, se hacen posibles cosas tan interesantes como el *control de versiones* o la compilación de documentos basada en *dependencias*. El primero de ellos permite *viajar al pasado* y recuperar estados anteriores de un escrito. El segundo es la piedra angular de la eficiencia: si un archivo cambia (un gráfico, por ejemplo), el documento que lo contiene se *imprime* de nuevo (de lo contrario, no se consume ni tiempo ni energía). Es decir, se pasa de concebir el escrito científico como un lienzo donde simplemente se depositan palabras, dibujos y datos tabulados, etc., a verlo más bien como un *proyecto vivo* donde sus distintos componentes se interrelacionan y procesan en cadena para constituir la publicación final. Esto posibilita, a su vez, la cooperación con terceros autores (compañeros de clase, miembros del grupo de investigación, etc.), lo cual es uno de los cimientos de la Física.

En este contexto, se pueden usar modernas herramientas de construcción de proyectos de *software* como el archiconocido GNU Make [19] u otras más modernas como Just, Ninja o Meson. Latexmk y Rubber son dos utilidades parecidas, pero más enfocadas a LaTeX. Lo bonito de estos flujos de trabajo es que, en lugar de producirse programas (ejecutables), obtenemos documentos listos para imprimir. Por ejemplo, el siguiente código GNU Make muestra cómo *construir* con Rubber el documento (documento.pdf) a partir de su fuente (documento.tex) y de una figura (diagrama.dia), diseñada en el *software* de dibujo Dia:

```
documento.pdf : documento.tex diagrama.tex
        rubber -d documento
diagrama.tex : diagrama.dia
        dia -e $@ -t tex diagrama.dia
```

Más tareas literarias y reactivas

La combinación de *notebooks* computacionales y explicaciones explorables ha transformado notablemente la manera en que se aborda la enseñanza y la comunicación de conceptos técnicos. Los *notebooks* computacionales, como Jupyter, ofrecen un entorno interactivo que integra código, texto y visualizaciones en una plataforma única. Esto facilita la creación de explicaciones explorables, donde los estudiantes pueden modificar directamente el código y ver cómo afecta a los resultados de manera inmediata. Los *notebooks* computacionales permiten una ejecución de actividades transparente y reproducible, lo que es esencial para la investigación, la colaboración y el acercamiento de los alumnos a la realidad académica y profesional.

¿El fin de LaTeX?

Anteriormente hemos abogado la suplantación parcial de LaTeX para la redacción y composición de determinados textos, tales como pequeños apuntes, entregas de ejercicios, etc., ya que tiende a producirse el mencionado *efecto Lamborghini*. Sin embargo, para la elaboración de documentos más complejos en Física (*preprints*, tesis, trabajos de fin de grado, etc.), LaTeX sigue siendo el indiscutible rey. Dicho esto, esta dinastía a veces parece sostenerse sobre continuos *parches* a la obra de Knuth y Lamport para intentar adaptarla a los tiempos.

Quizás, el más significativo haya sido XeTeX [20], que posibilita el uso de codificación Unicode y tipos de letra modernos (OpenType, TrueType, etc.).

Es posible que a LaTeX le haya salido un serio competidor. Typst [22] es un sistema de composición de textos que tiene varias ventajas sobre LaTeX. La más importante es que su lenguaje de marcado es mucho más sencillo, lógico, versátil y limpio (fig. 9) y su compilador (el *traductor* a PDF) está desarrollado con herramientas de programación modernas [23].

Conclusiones

En este artículo hemos tratado la evolución de los sistemas de composición de textos en el ámbito de la enseñanza y la divulgación de la Física. Se ha destacado que, a pesar del avance de las tecnologías interactivas, el medio principal para transmitir conocimiento en Física sigue siendo el texto escrito. Se ha reivindicado TeX/LaTeX como un sistema revolucionario de edición que lleva permitiendo la creación de documentos científicos de alta calidad desde hace varias décadas. También se han presentado otros sistemas y herramientas relacionadas, como LyX, MathML, Markdown y entornos de programación letrada, como los *notebooks* computacionales. Hemos señalado también la importancia de las tareas explorables, así como el uso de la nube para la distribución de contenidos educativos. Por último, hemos introducido Typst como un flamante competidor a LaTeX.

Desde nuestra experiencia, recomendamos a los profesores de Física en enseñanzas secundarias el incidir en la separación entre estilo y contenido, en la importancia del texto plano y en el uso de lenguajes de marcado ligero para realizar ejercicios, presentaciones, monografías, etc. Estos conceptos son muy importantes y representan sin duda la base sobre la que aplicar el resto de las tecnologías discutidas (TeX, notebooks, web, Typst, etc.) u otras venideras. Por último, sugerimos el uso de servicios *online/cloud* siempre que sea posible para facilitar la movilidad, la colaboración y la sencillez para el estudiante.

Referencias

[1] D. Knuth, *The TeXbook* (Addison-Wesley Reading, MA, 1986).

[2] U. Quill, Introduction to LyX: Make working with LaTeX easier using the WYSIWYG editor LyX, *Linux Journal* **1999**, no. 57es, pp. 6–es, (1999).

[3] R. Miner, The importance of MathML to mathematics communication, *Notices of the AMS* **52** (5), 532 (2005).

[4] H. Ferreira y D. Freitas, AudioMath: Using MathML for speaking mathematics, *XATA05 - XML and Associated Technologies* (Universidade do Minho, Braga, Portugal, 2005).

[5] D. Cervone, P. Krautzberger y V. Sorge, Towards universal rendering in MathJax, *Proceedings of the 13th international web for all conference*, p. 1 (2016).

[6] T. Mailund, *Introducing Markdown and Pandoc*, vol. 1 (Apress, 2019).

[7] E. Schulte y D. Davison, Active documents with Org Mode, *Computing in Science & Engineering* **13** (3), 66 (2011).

[8] D. E. knuth, Literate programming, *The Computer Journal* **27**(2), 97 (1984).

[9] S. Wolfram, *The Mathematica book*, vol. 1 (Wolfram Research, 2003).

[10] J. Somers, The scientific paper is obsolete, *The Atlantic*, 5 abril (2018).

[11] C. Boettiger, An introduction to Docker for reproducible research, *ACM SIGOPS Operating Systems Review* **49**(1), 71 (2015).

[12] A. Corbi, D. Burgos y A. M. Pérez, Cloud-operated open literate educational resources: The case of the MyBinder, *IEEE Transactions on Learning Technologies* **17**, 893 (2023).

[13] M. Beg *et al.*, Using Jupyter for reproducible scientific workflows, *Computing in Science & Engineering* **23**(2), 36 (2021).

[14] S. Mati, I. Civcir y S. Abba, EviewsR: an R Package for Dynamic and Reproducible Research Using EViews, R, R Markdown and Quarto, *R Journal* **15**(2), 169 (2023).

[15] Y. Xie, knitr: a comprehensive tool for reproducible research in R, *Implementing reproducible research*, p.3 (Chapman; Hall/CRC, 2018).

[16] A. Zúñiga-López, C. Avilés-Cruz, A. Ferreyra-Ramírez y E. Rodríguez-Martínez, Jupyter-notebook: a digital signal processing course enriched through the Octave Programming Language, *Intelligent computing: Proceedings of the 2020 computing conference,* vol. 1, p. 774 (Springer, 2020).

[17] B. Victor, Humane representation of thought: A trail map for the 21st century, *Proceedings of the companion publication of the 2014 ACM SIGPLAN conference on systems, programming, and applications: Software for humanity*, p. 5 (2014).

[18] J. M. Perkel, Reactive, reproducible, collaborative: Computational notebooks evolve, *Nature* **593**, 156 (2021).

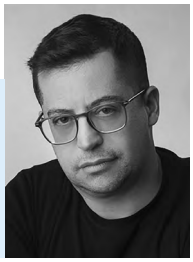
[19] R. Mecklenburg, *Managing Projects with GNU Make* (O'Reilly Media, 2004).

[20] J. Kew, About XeTeX, *University of Utah*, vol. 5 (2005).

[21] L. Mädje, A programmable markup language for typesetting, PhD thesis, TU-Berlin (2022).

[22] M. E. Haug, Fast typesetting with incremental compilation, PhD thesis, TU-Berlin (2022).

[23] N. D. Matsakis y F. S. Klock, The Rust language, *ACM SIGAda Ada Letters* **34**(3), 103 (2014).



Alberto Corbi
Instituto de Investigación, Innovación y Tecnología Educativas (UNIR iTED), Universidad Internacional de la Rioja (UNIR)